PUC-Rio – Software Básico – INF1018
Prova 2 – Turma 3WA – 01/12/2016

1. (2,5 pontos)

```
#include <stdio.h>

void dump (void *p, int n) {
   unsigned char *p1 = (unsigned char *) p;
   while (n--) {
      printf("%p - %02x\n", p1, *p1);
      p1++;
   }
}

struct X {
   short s;
   double d;
   float f;
  } x = {-3, -2.625, 530.0};

int main (void) {
   dump (&x, sizeof(struct X));
   printf ("\n%d\n", sizeof(x));
   printf ("\n%d\n", sizeof(struct X));
   return 0;
}
```

Supondo que x seja armazenado no endereço de memória 0x8049600, diga o que o programa irá imprimir quando executado, deixando claro como você chegou a esses valores. Considere que a máquina de execução é *little-endian*, e que as convenções de alinhamento são as do Linux no IA-32. Se houver posições de *padding*, indique seu conteúdo com **PP**. (ATENÇÃO: valores sem contas e explicações **NÃO** valem ponto!)

2. (1,5 pontos) Considere o seguinte código em C:

```
#include <stdio.h>
extern float b;
float a = 5.0;

float altera(float v);

float final() {
   float c;
   c = altera(b) * a;
   return c;
}

int main(void) {
   float f = final();
   printf("%f\n", f);

   return 0;
}
```

Liste quais símbolos do módulo objeto do código acima seriam importados e exportados, ou seja apareceriam como **D** (símbolo na área de dados), **T** (símbolo na área de código) e **U** (símbolo indefinido) na saída do programa `nm`.

3. (2,5 pontos) Traduza a função `calc` abaixo para assembly IA-64 (o assembly visto em sala), utilizando as regras usuais de alinhamento, passagem de parâmetros e resultados de C/Linux. Traduza o mais diretamente possível o código de C para assembly. Comente seu código.

```
double floor(double v);

double calc (int n, double *vals, double x) {
   double total = 0.0; int i;

   for (i=0; i < n; i++) {
      total += (x - vals[i]);
   }
   return floor(total/n);
}
```

4. (2,5 pontos) Implemente em assembly a função

```
void converte (float val, double* novo_val);
```

que converte um número ponto flutuante de precisão simples para um número de precisão dupla. Essa função deve ser implementada manipulando apenas a representação em memória dos números em ponto flutuante.

5. (1,0 pontos) Suponha que compilamos e ligamos um determinado programa `chama.c` onde a `main` contém uma chamada a uma função `moo`, gerando um executável `chama`. Ao executar `objdump -d chama`, um dos trechos resultantes é o seguinte (com uma pequena modificação explicada a seguir):

```
Endereço        código de máquina       código assembly

08048374 <moo>:
 8048374:       55                      push   %ebp
 8048375:       89 e5                   mov    %esp,%ebp
 8048377:       b8 01 00 00 00          mov    $0x1,%eax
 804837c:       5d                      pop    %ebp
 804837d:       c3                      ret

0804837e <main>:
 804837e:       8d 4c 24 04             lea    0x4(%esp),%ecx
 8048382:       83 e4 f0                and    $0xfffffff0,%esp
 8048385:       ff 71 fc                pushl  0xfffffffc(%ecx)
 8048388:       55                      push   %ebp
 8048389:       89 e5                   mov    %esp,%ebp
 804838b:       51                      push   %ecx
 804838c:       e8 HH HH HH HH          call   <moo>
 8048391:       59                      pop    %ecx
 8048392:       5d                      pop    %ebp
 8048393:       8d 61 fc                lea    0xfffffffc(%ecx),%esp
 8048396:       c3                      ret
```

Neste trecho, substituímos uma sequência de quatro bytes originalmente exibidos em hexadecimal pela sequência "HH HH HH HH". Calcule os valores hexadecimais da sequência original.

Boa Prova!